Dhruv Singal

CountAnalysis

Since we are using an enhanced MergeSort, the recurrence relation is very similar.

Problem size: $n \leftarrow$ # elements in array provided as input

Basic operation: key comparison

Worst case: data entries are interleaved such that at the final merge, the left section & right section have interleaved ascending values.

eg: $\underbrace{1, 3, 5}_{A}$ merge with $\underbrace{2, 4, 6}_{B}$

requires n - comparisons

Assumption n is a power of 2

ie $\log_2 n \in \mathbb{N}$

Total # comparisons: $= M(n)$

∵ The function mergeSortCountInversion recurses halving the problem each time, and the function mergeSortedCountSplitInversions does n comparisons at each step, where n = size of left array + size of right array.

$\therefore M(n) = \underbrace{M\left(\frac{n}{2}\right)}_{\text{left}} + \underbrace{M\left(\frac{n}{2}\right)}_{\text{right}} + \underbrace{n}_{\text{worst case \# comparisons}}$

$= 2M\left(\frac{n}{2}\right) + n$

$$= 2\left(2M\left(\frac{n}{4}\right) + \frac{n}{2}\right) + n$$

$$= 4M\left(\frac{n}{4}\right) + 2n$$

$$= 8M\left(\frac{n}{8}\right) + 3n$$

$$\vdots$$

$$= 2^x M\left(\frac{n}{2^x}\right) + nx$$

$n = 2^x$
$x = \log_2 n$

$$= n\,M(1) + n\log_2 n$$

$$= n\log_2 n \quad (\because M(1) = 0) \text{ Base case}$$

∴ total # key comparisons = $n\log_2 n$

∴ closed form solution:

$$\boxed{T(n) = n\log_2 n}$$